REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JANUARY 2010	Conference Paper Postprint	February 2006 – December 2007

4. TITLE AND SUBTITLE

IMPROVING TIMELINESS AND RELIABILITY OF DATA DELIVERY IN TACTICAL WIRELESS ENVIRONMENTS WITH MOCKETS COMMUNICATIONS LIBRARY

5a. CONTRACT NUMBER
FA8750-06-2-0064

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)

Erika Benvegnu, Niranjan Suri, James Hanna, Vaughn Combs, Robert Winkler, and Jesse Kovach

5d. PROJECT NUMBER
ICED

5e. TASK NUMBER
06

5f. WORK UNIT NUMBER
02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

AFRL/RISE
525 Brooks Road
Rome, NY 13441-4505

Florida Institute for Human and Machine Cognition
40 South Alcaniz Street
Pensacola, FL 32502-6008

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISE
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2010-2

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited PA# 88ABW-2009-5306 Date Cleared: January 4, 2010

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Network centric warfare relies on the timely and reliable delivery of data to disparate cooperating nodes in tactical networking environments. Given the limited bandwidth available and the unreliability of network links, data often accumulates in application and/or network queues, resulting in increased latency in the delivery of the data. The Mockets communications library addresses this problem via dynamic message replacement. The message replacement functionality of Mockets allows the system to drop all but the most recent message within a specific message flow by removing older, outdated messages from the queues. This paper describes and evaluates, in the context of the U.S. Air Force's Joint Battlespace Infosphere (JBI) system, the timeliness of end-to-end delivery of data using the Mockets library. In addition to dynamic message replacement, other capabilities in the Mockets library include options for reliable vs. unreliable and sequenced vs. unsequenced delivery of data, detailed statistics and feedback regarding the connection, and assignment and dynamic adjustment of priorities of messages. This paper provides a qualitative analysis of these different capabilities and their suitability to address the transport requirements in JBI. It also provides a quantitative comparison of Mockets with SCTP and SCPS-TP, which are similar technologies with existing available candidate implementations. Our results show that the Mockets library with the message replacement significantly outperforms these other transport protocols.

15. SUBJECT TERMS
Mockets, Publish/Subscribe Middleware, Information Management

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			James Hanna
U	U	U	UU	9	19b. TELEPHONE NUMBER (*Include area code*) N/A

IMPROVING TIMELINESS AND RELIABILITY OF DATA DELIVERY IN TACTICAL WIRELESS ENVIRONMENTS WITH MOCKETS COMMUNICATIONS LIBRARY

Erika Benvegnù, Niranjan Suri
Florida Institute for Human & Machine Cognition, Pensacola, FL, USA

James Hanna, Vaughn Combs
U.S. Air Force Research Laboratory, Rome, New York, USA

Robert Winkler, Jesse Kovach
U.S. Army Research Laboratory, Adelphi, Maryland, USA

ABSTRACT

Network centric warfare relies on the timely and reliable delivery of data to disparate cooperating nodes in tactical networking environments. Given the limited bandwidth available and the unreliability of network links, data often accumulates in application and/or network queues, resulting in increased latency in the delivery of the data. The Mockets communications library addresses this problem via dynamic message replacement. The message replacement functionality of Mockets allows the system to drop all but the most recent message within a specific message flow by removing older, outdated messages from the queues. This paper describes and evaluates, in the context of the U.S. Air Force's Joint Battlespace Infosphere (JBI) system, the timeliness of end-to-end delivery of data using the Mockets library.

In addition to dynamic message replacement, other capabilities in the Mockets library include options for reliable vs. unreliable and sequenced vs. unsequenced delivery of data, detailed statistics and feedback regarding the connection, and assignment and dynamic adjustment of priorities of messages. This paper provides a qualitative analysis of these different capabilities and their suitability to address the transport requirements in JBI. It also provides a quantitative comparison of Mockets with SCTP and SCPS-TP, which are similar technologies with existing available candidate implementations. Our results show that the Mockets library with the message replacement significantly outperforms these other transport protocols.

INTRODUCTION

Tactical networking environments are often characterized by limited bandwidth and unreliable connectivity due to their wireless ad-hoc nature. Nevertheless these environments must still support applications with potentially massive data exchange requirements.

Our analysis focuses on the context of the U.S. Air Force's Joint Battlespace Infosphere (JBI) in order to find a transport protocol that supports transmission and reception of data between clients and servers and provides advanced functionalities to be exploited by the infosphere. In this analysis the main aim of the transport protocol is to maximize the timeliness of the delivery of information, as well as providing advanced capabilities such as supporting different types of flow, integrating with other components, and controlling the communication in a flexible manner.

In the JBI system, several applications have timing constraints on the delivery of data. Information may become stale after some amount of time. Additionally, some old messages become irrelevant when a newer message becomes available. Message replacement allows newer information to be delivered quickly, by removing old messages from the transmission queue that are either waiting to be sent or are awaiting retransmission if not acknowledged.

Our evaluation consisted of identifying the most suitable transport technologies to support JBI applications with timeliness constraints followed by a quantitative comparison of those technologies.

In our quantitative analysis we have considered several types of applications in different network environments. A series of tests has been performed analyzing applications involving both single and multiple information producers and also including several different types of information with different delivery requirements. During the experimentation we simulated two different environments: a low bandwidth channel and a channel with fluctuating bandwidth.

JBI AND THE REQUIREMENTS OF THIS CONTEXT

JBI is an information management architecture that supports efficient publish, subscribe, and query of information [1] [1]. JBI was initially targeted towards enterprise networks such as those present at Air Operations Centers and other upper echelons, but has since been extended to support airborne networking environments. In the context of enterprise networks, JBI uses Java Remote Method Invocation (RMI) and the Java Messaging Service as transport protocols. However, the airborne networking environment introduces low-

978-1-4244-5239-2/09/$26.00 ©2009 IEEE

bandwidth, unreliable tactical radio links, which require specialized support from transport protocols.

The primary goal of a transport protocol in the airborne networking context is to maximize the timeliness of delivery of data to the information space and to clients, while minimizing the overhead in providing this delivery capability. In addition to being efficient, the transport protocol should provide reliable and/or sequenced delivery of data, as required by the semantics of the communication. Moreover, the transport capability should support tight integration with other components that are managing the information dissemination. Information and statistics available at the transport layer should be exposed to the information management system and client runtime system. The transport layer should also provide flexible control interfaces to allow the other components to customize the behavior and operation of the transport layer. For example, the transport layer should allow readjustment of priorities of messages, and creation of multiple streams for traffic differentiation.

THE MOCKETS COMMUNICATION LIBRARY

Mockets (mobile sockets) is a communication library specifically designed to improve communications in MANET environments [1]. Mockets offers several delivery services and applications can choose independently among reliable or unreliable, sequenced or unsequenced delivery of data. Different delivery services can coexist in the same connection.

Another feature of Mockets is the network condition monitoring component [4]. This component makes the current network state available to the application. The network condition is measured in terms of available bandwidth along the communication path, round trip time, packet loss rate, and peer reachability. This allows an application using Mockets to adapt to changes in the metwork environment by tuning several communication parameters. Applications can dynamically change the priority and maximum lifetime of messages. They can also set an enqueue timeout for the insertion of messages in the transmission queue and a retry timeout that specifies for how long a message will be retransmitted before being discarded. Applications can tune the parameters for each message or for a specific flow of messages.

Mockets also supports message tagging, cancellation, and replacement. By tagging messages, applications can separate message flows and perform group operations on a specific type of messages. Tagging messages also allows applications to use the functionalities of message cancellation and replacement. An application that faces a congested network situation or limited bandwidth can decide to cancel all the messages of a specific type. A call to cancel will cause all the messages marked with the specified tag to be deleted from the transmit queues. The message replacement functionality consists of a cancel followed by a new send. This capability allows applications to deliver only the most recent update of a specific flow. When a new message is available, using the replacement will cause the messages marked with a specific tag to be deleted from the transmit queues and the new message to be enqueued. This technique reduces the message latency by dropping the transmission rate. Message replacement was identified as the feature that would most benefit applications with timeliness constraints in the JBI system.

In addition, in recent months the Air Force Research Laboratory (AFRL) has developed a reference set of Information Management (IM) Services that will provide an essential piece of the envisioned final Net-Centric IM solution for the Department of Defense (DoD). The IM Services will provide mission critical functionality to enable seamless interoperability between existing and future DoD systems and services while maintaining a highly available IM capability across the wide spectrum of differing scalability and performance requirements. The services developed using the Phoenix architecture will provide capabilities for information submission, information brokering and discovery, repository, query, type management, dissemination, session management, authorization, service brokering, and event notification [5]. In addition, the IM services support common information models that facilitate the management and dissemination of information consistent with client needs and established policy. The services support flexible and extensible definitions of session, service, and channel contexts that enable the application of Quality of Service (QoS) and security policies at many levels within the SOA.

The Phoenix architecture provides information and byte channels for the transmission of all information between and among clients and services. In addition, a channel context is associated with each channel in order to enable control and policy enforcement components to both instrument and affect channel behaviors dynamically. In the current implementation the Mockets library has been used as one of the underlying enabling byte channel technologies. By simply setting appropriate channel context attributes the underlying Mockets library can be used to choose among reliable or unreliable, sequenced or unsequenced delivery of information. In addition, context attributes may be changed to impact information delivery

priority and information replacement based on QoS or other mission priorities and policies.

QUALITATIVE ANALYSIS

We identified a variety of technologies which might be suitable for use in the JBI context.

- UDP/TCP (User Datagram Protocol and Transmission Control Protocol)
- SCTP (Stream Control Transmission Protocol)
- RUDP (Reliable UDP)
- DCCP (Datagram Congestion Control Protocol)
- SCPS-TP (Space Communications Protocol Standards – Transport Protocol)
- Mockets communication library
- DDS (Data Distribution Service)
- JMS (Java Message Service)

These technologies can be divided into transport protocols and publish/subscribe middleware.

Transport Protocols:

UDP and TCP are core protocols of the Internet Protocol suite. UDP offers an unreliable, unsequenced delivery that does not meet the basic requirements of JBI. TCP instead offers a reliable, in-sequence delivery, but was designed for fully connected wired infrastructure environments and exhibits several weaknesses in wireless, low bandwidth, intermittently connected environments.

SCTP [6] is a transport layer protocol developed to improve and add functionalities to TCP. SCTP ensures reliable, in-sequence transport of messages with congestion control (like TCP). However message ordering is optional, the user application can process messages in the order they are received. SCTP supports multiple streams of messages. Streams can be exploited, for example, to accord higher priority to control messages over data messages, however inter-stream priority is currently not a standard feature. SCTP also offers a path selection-and-monitoring feature that allows selecting a “primary” data transmission path and testing the connectivity of the transmission path.

RUDP extends UDP by adding acknowledgments and retransmission of lost packets, windowing and congestion control. It offers a reliable, in-order delivery [7]. RUDP's congestion control mechanisms allow streams to behave in a TCP-friendly fashion without disturbing the real-time nature of the protocol. Even though RUDP offers a reliable delivery it does not support advanced features like prioritizing messages and multiple streams that are required for JBI.

DCCP is a message-oriented transport layer protocol. It implements reliable connection setup and teardown as well as several congestion control mechanisms [8]. The protocol provides optional mechanisms to allow an application to set the reliability of the transmission according to its needs. DCCP can tell the sender application with high reliability as to which data packets reached the receiver and whether these packets were corrupted, or dropped in the receive buffer. However reliability has to be implemented on top of DCCP, it is not included in the protocol. This communication protocol is suitable for applications that can benefit from control over the tradeoff between timeliness and reliability but reliability is a requirement for the applications we are focusing on in this discussion.

SCPS is a protocol suite designed to support communications over challenging environments. We focused on SCPS-TP, an underlying transport protocol optimized to provide reliable end-to-end delivery of messages between hosts communicating through a satellite link. This protocol was designed to meet the needs of satellite communications but it has been proven to work well also for mobile/wireless and tactical communications [9]. SCPS-TP offers reliable delivery (complete, correct, in sequence, with no duplications), a best effort delivery (correct, in sequence, no duplications, possibly with gaps) and a delivery with a minimal reliability (correct, possibly incomplete, possibly out of sequence). SCPS also supports message priorities.

As already described, the Mockets library is a communication library specifically designed to improve communications in wireless networking scenarios. Mockets provide several delivery services with different communication semantics: reliable/unreliable, sequenced/unsequenced. Mockets allows applications to perform semantic classification of data, cancellation and replacement of enqueued data. It also supplies several fine-tuning mechanisms like message priority and timeouts. Mockets also provides a network monitoring component that can assist applications in dynamically adapting to changes in the communications infrastructure.

Publish/Subscribe Middleware:

DDS [10] is an implementation of a data-centric publish/subscribe middleware for distributed systems. DDS is implemented as a decentralized peer-to-peer architecture that allows the user to specify QoS parameters such as best effort or reliable and sequenced delivery of

data, priority of messages, and grouping of data (creating multiple streams). It also allows monitoring the connectivity.

JMS is a Java Message Oriented Middleware API for sending messages between clients. JMS supports two models: a point-to-point model where only one consumer will get the message and then send back an acknowledgement (reliable service) and a publish/subscribe model. JMS enables distributed communication that is loosely coupled, reliable, and asynchronous [11].

From the qualitative examination we decided that the transport protocols are more suitable as publish/subscribe middleware does not seem to satisfy the specific necessities of JBI. Publish/subscribe middleware might be able to perform reliable service and prioritizing messages, like in the case of DDS, but they are not meant to be used for point-to-point communication. Moreover the effort to adapt the middleware to the environment and get the needed capabilities will reduce the middleware performance. Among the transport protocols, we also ruled out UDP and DCCP because they do not offer a reliable service that is a prerequisite in the target environment. We also ruled out also TCP because of its well-known weaknesses when working in a wireless network. Finally, RUDP was ruled out because it does not have a standard implementation and it does not offer the required capabilities of prioritization and multiple streams.

Table 1 specifies various characteristics of transport protocols, some of which are necessary to meet the requirements of the JBI environment and others that are useful to improve performance. The table summarizes the features available for each of the technologies.

Table 1: Transport Protocols and their Features

	UDP	TCP	SCTP	DCCP	RUDP	SCPS-TP	Mockets
Reliable transport		✔	✔		✔	✔	✔
Ordered delivery		✔	✔	✔	✔	✔	✔
Unordered delivery	✔		✔			✔	✔
Congestion control		✔	✔	✔	✔	✔	✔
Multiple streams			✔	✔		✔	✔
Priority			✔			✔	✔
Connection monitoring			✔				✔

QUANTITATIVE ANALYSIS

Following the qualitative downselect, we decided to perform a quantitative evaluation to compare the surviving technologies: SCTP, SCPS-TP and Mockets. The experimental evaluation consists of four tests all designed to evaluate the timeliness of information delivery.

The experiments performed simulate a scenario involving blue force tracking with the JBI system. We measure the timeliness of data delivery and highlight the ability of Mockets to provide good performance in this context, taking advantage of its unique message replacement capability.

Experiment One

In the first test case, a client connects over a low-bandwidth link to the JBI system that is disseminating blue force tracking information for all the nodes that are in theatre. This particular configuration assumes 20 nodes, each generating an update message at a rate of 1 Hz. Each message is assumed to be 1 KB in size. The JBI system reflects these messages back to all the clients, which requires a bandwidth of 20 KB/s between the JBI system and each client.

The disadvantaged client, however, does not have a 20 Kb/s communications channel available. We limit the bandwidth to 5 KB/s, which is usually the maximum bandwidth available for a tactical radio link such as a PRC-117a radio or a PSC-5 ground terminal. Due to the limited link bandwidth the messages will accumulate in the network queues because the system produces more messages than it is able to send out. Messages residing in the queues cause delay in the delivery. Moreover when the queues are full messages are dropped and they have to be retransmitted (due to reliable service) causing additional traffic. We assume a queue of 4 packets (4 KB) in the network card. In this test we can appreciate how the message replacement function of Mockets is able to replace packets that have not yet been sent, or are waiting for acknowledgement, with a newer update as soon as this is produced. This function allows the sender to reduce the number of packets injected in the congested network and allows the receiver to get the newer position update instead of receiving all the updates with a considerable delay.

The congestion control of SCTP and SCPS-TP helps reducing the sending rate, but this does not avoid the problem of getting the updates with a considerable delay since the rate at which the updates are produced does not decrease and the bandwidth of the channel does not increase overtime.

SCPS-TP offers several congestion control mechanisms. We decided to perform the tests with the standard mechanism (TCP Vegas style congestion control) given that preliminary tests showed that the variation in performance was negligible across the different algorithms.

Figure 1 shows the result of the first experiment. The latency is evaluated as the time elapsed from when the sensor update was produced and when it is received from the listening application.

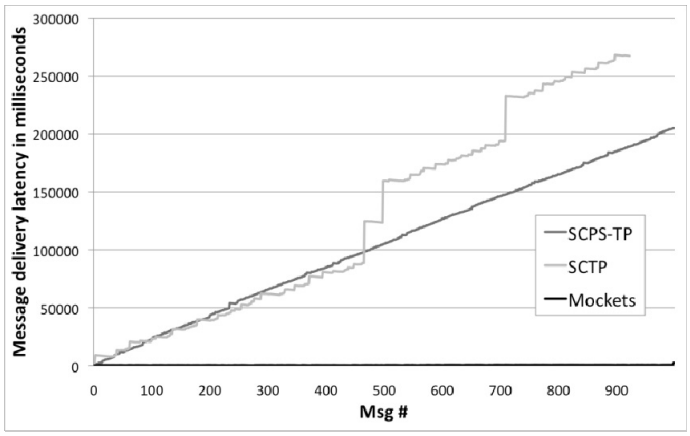

Figure 1: Latency comparison between SCPS-TP, SCTP and Mockets

The message replacement allows Mockets to drastically reduce the sending rate by sending only the most recent update while old messages waiting in queue are cancelled. The average latency using Mockets is 528 ms whereas with SCPS-TP and SCTP we have an unbounded growth in the latency. We can see how the congestion control mechanism used by SCTP is very aggressive, causing a drastic delay of the message delivery since the sending rate of the application is not reduced. Note that the performance of Mockets is represented by a black line that is almost directly over the horizontal axis, making it difficult to see.

Figure 2 is a close-up of Figure 1 showing the latency for the first 100 messages sent. This chart highlights the messages the client application actually received with markers. This clearly shows that the performance improvement achieved by Mockets are not due to a higher throughput. Using Mockets the application sends 283 messages out of 1000 produced by the sensor, 25 out of 100. This result is reasonable given that the available bandwidth is 5KB/s 25% of the producing rate, which is 20KB/s, and the amount of data sent using Mockets is 28.3% of the total generated data.

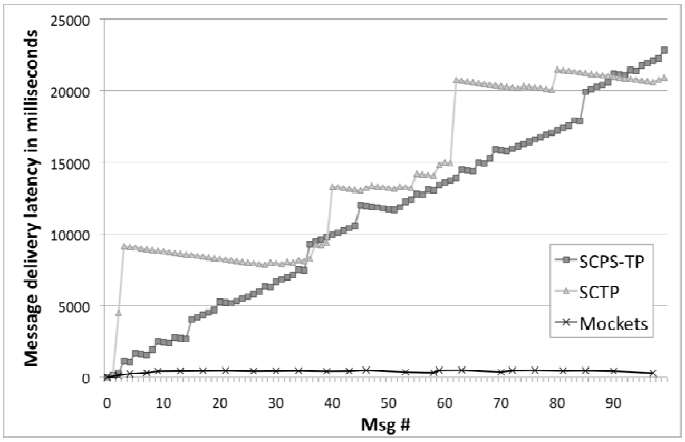

Figure 2: Latency comparison: close-up of the first 100 messages.

Experiment Two

In the second experiment, a second data stream, consisting of sensor data, is introduced in addition to the blue force tracking data. For simplicity, we assume the sensor data stream also consists of 1 KB messages at an average 20 Hz rate. The network bandwidth for the disadvantaged client remains the same 5 KB/s as in the first experiment. Given the two streams, the load on the network will be heavier and the delay of each message will be longer. This test evaluates the Mockets message replacement working in combination with the message tagging functionality in order to replace the right message among several streams, maintaining a small, constant delay for messages across different streams.

Figure 3 shows a close-up of the result of this experiment, displaying the latency of the first 100 messages. The figure shows two data series for each technology: one data series for each of the two streams, identified as Type A and Type B. Using SCPS-TP and SCTP, the latency of each message is about three times the one measured in the first test. Message 100 of SCPS-TP Type A has a latency of 63486 ms, about 2.7 times the latency of message 100 of SCPS-TP in the first test which is 22782 ms, similar proportions can be found throughout the results. A rate of increase of about three times is reasonable since the sending rate is double the one of the first test and the queues are full so more messages get dropped creating even more traffic since the messages are delivered reliably and therefore each dropped message has to be sent again.

With Mockets, different tags are assigned to messages from stream A and from stream B, so that when a new update is available, only messages of the same flow are replaced. The average latency of messages sent using Mockets is 488 ms for Type A and 118 ms for Type B.

The different latency is coupled with a different number of messages delivered: 152 out a 1000 for Type A and 87 out of 1000 for Type B. This behavior is explainable considering that stream A (blue force tracking) starts first, so the initial messages of Type A that fill the queue are more likely to go through. Note that since fewer messages of Type B are sent, the latency for this type of message is lower.

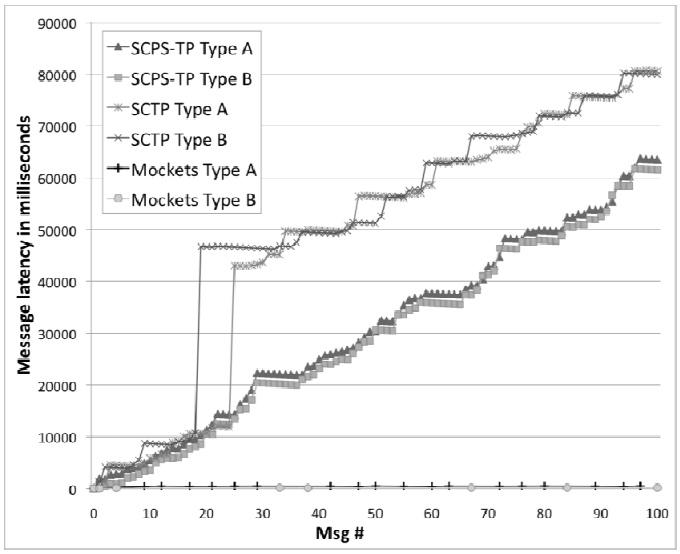

Figure 3: Close-up of the latency of the first 100 messages when sending two flows of position updates.

Experiment Three

In the third experiment, we introduce different priorities for the different types of information sent over the low bandwidth channel between the JBI system and the disadvantaged client. We have position updates as before for blue force tracking, and images from a sensor. As described earlier, the position updates are frequent, small messages of size 1 KB at a rate of 20 Hz. The images sensor on the other hand produces updates of 5KB once every 10 seconds. These sensor data messages are not subject to replacement as every single message is important. Hence, they are sent using a reliable, unsequenced, high priority stream.

Figure 4 shows the results of the third experiment. The series labeled Type A represents position update messages, and Type B represents image updates from a sensor. The average latency for Type A using Mockets is 773 ms, slightly higher than the latency with just position updates. Also the latency of each message of Type A using SCPS-TP and SCTP is slightly higher than in the first experiment. Over the time to produce one thousand position updates only six image updates has been produced, so the overhead on the congested network is not significant. Image updates, type B, suffer from little latency since they are higher priority messages. The latency for updates of Type B is 1661 ms using SCPS-TP, 1391 ms using SCTP and 3471 ms using Mockets. Note that messages of Type B are also unsequenced, and the higher latency using Mockets is due to message number 4 that got lost and arrived out of order, message 5 and 6 arrived before message 4.

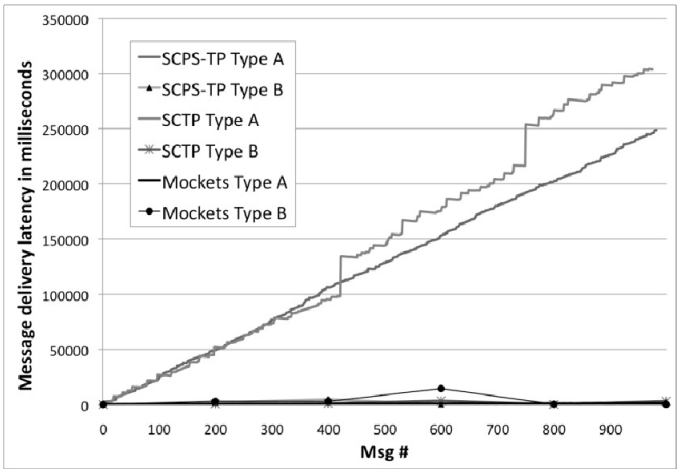

Figure 4: Multiple types of flow from two different sensors

Experiment Four

To perform the fourth and last experiment we simulated a link with a fluctuating bandwidth, normally at 40 KB/s but periodically dropping to 5 KB/s. The bandwidth of the channel is 40 KB/s for 50 seconds then it is reduced to 5 KB/s for 10 seconds. We assumed ten nodes (instead of twenty), each generating the same 1 KB position update at a rate of 1 Hz. In this configuration, when the link bandwidth is 40 KB/sec, there is more than adequate bandwidth for all the messages to be transmitted. When the bandwidth drops the sending rate is double the capacity of the channel.

Figure 5 shows clearly how the performance of SCPS-TP and SCTP are quickly deteriorated even with a small period of time of limited bandwidth, to the point where SCTP could not recover from the small period of limited bandwidth. On the other side the Mockets message replacement starts to work immediately by replacing stalled messages with newer updates and delivering them with very little latency and is prompt to stop replacing messages as soon as the bandwidth is restored.

This experiment has been performed five times, registering an average latency 1012 ms using SCPS-TP, 10998 ms using SCTP and 40 ms using Mockets.

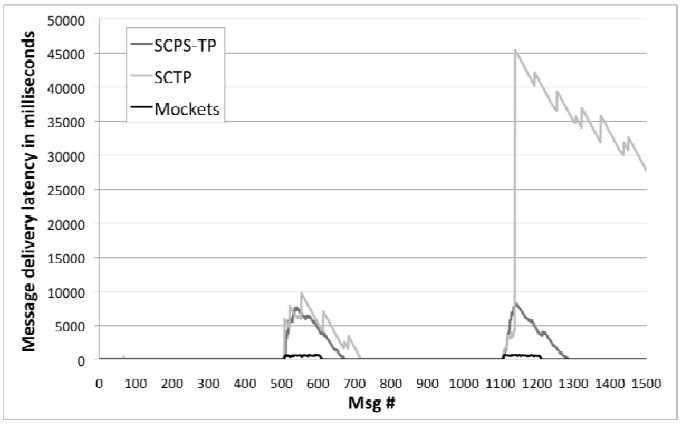

Figure 5: Latency comparison with fluctuating bandwidth. First run of experiment four.

Discussion

The issues that applications with liveliness constraints face are not only due to the delay of the single message, but also due to the fact that when a new update is available, older updates become stale. Using the Mockets message replacement, we sacrifice completeness in order to have prompt delivery of newer updates. The message stream is reliable, sequenced because the application is always interested in at least the most recent message. Using other technologies we could obtain a more timely distribution with unreliable delivery given that reliable, in-order delivery combined with congestion control creates considerable traffic and therefore considerable delay. An unreliable flow would help in delivering with less delay but would not ensure the liveliness of data, i.e. messages would have less delay but a newer update that overrides old messages may be available but lost in transmission. Also consider this example: an application sends updates at irregular intervals of time: it sends M_1 at time t_1, M_2 at time t_2 and M_3 at time t_3. The interval between t_1, t_2 and t_3 happens to be very long and M_2 is lost. If we are using unreliable delivery the receiver application would have as last update M_1 for a long time, until M_3 is received (t_3-t_1), while a newer update was available but has not been delivered, the information M_1 is out of date from t_2 to t_3. What Mockets achieves with message replacement is replacing a message only when a newer one is available. The newest message is delivered reliably so if it gets lost it is resent until it is successfully delivered or until a newer update comes available. Note how this mechanism reduces the sending rate and the load on the congested network while maintaining liveliness and timeliness of data.

Experimental Setup Details

The tests have been performed using the most recent release of Mockets as of April 2009, SCPS Reference Implementation 1.1.13 software distribution, and the Linux Kernel implementation of SCTP, version 1.0.9.

The experiments have been setup using two machines in the case of Mockets and SCPS-TP. These machines run Fedora Core 4 Linux. The machines are connected through a hub and both run NIST Net to emulate a low bandwidth radio link typical of the JBI system. For the SCTP tests we used two machines running Ubuntu 8.0.4 since this distribution includes a pre-packaged distribution of the Linux Kernel SCTP library in its repositories. The two machines were connected through a hub to a third machine emulating the low bandwidth connection using NIST Net. Two features of NIST Net have been used: the bandwidth limitation and the queue depth. When bandwidth limitation is used it causes packets to be queued. By default, NIST Net sets the queue depth to an infinite value which is not realistic. For our tests we set the queue depth to 4 packets, 4 KB in the case of 1KB packets, to simulate the queue length of a typical network card.

CONCLUSIONS

The message replacement capability of Mockets offers a clever way to enhance liveliness of messages and prompt delivery. Applications sending a flow of messages with time delivery constraint in a low bandwidth or congested network are the ones that benefit most from Mockets message replacement. Reliable delivery coupled with message replacement ensures the liveliness of data. Using the Mockets message replacement completeness of the information is sacrificed in order to achieve timeliness in delivery. In a situation with limited bandwidth, Mockets has a lower throughput than other technologies but the reduced amount of messages to be sent over the network, and selective cancellation of old messages from queues, allows Mockets to achieve good performance by delivering the most recent data in a timely fashion.

ACKNOWLEDGEMENTS

This work is supported in part by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-04-2-0013, by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0009, and by the Air Force Research Laboratory under Cooperative Agreement FA8750-06-2-0064.

REFERENCES

[1] Infospherics Web Site. Online reference: http://www.infospherics.org.

[2] M. Linderman, et. al., *A Reference Model for Information Management to Support Coalition Information Sharing Needs*, In Proceedings of 10th International Command and Control Research and Technology Symposium, 2005 http://www.dodccrp.org/events/10th_ICCRTS/CD/papers/274.pdf.

[3] N. Suri, M. Tortonesi, M. Arguedas, M. Breedy, M. Carvalho, R. Winkler, *Mockets: A Comprehensive Application-Level Communications Library*, in Proceedings of 24th Military Communications Conference (MILCOM 2005), Atlantic City, NJ, USA, October 2005.

[4] C. Stefanelli, M. Tortonesi, M. Carvalho, N. Suri, *Network Conditions Monitoring In The Mockets Communication Framework*, in Proceedings of 26th Military Communications Conference (MILCOM 2007), Orlando, FL, USA.

[5] R. Grant, V. Combs, J. Hanna, B. Lipa, J. Reilly, *Phoenix: SOA based information management services*, Proceedings of SPIE, Defense, Security, and Sensing, April 2009, Orlando, FL, USA

[6] R. Stewart, Q. Xie et. al., *Stream Control Transmission Protocol RFC 2960*, October 2000.

[7] D. Velten, R. Hinden, J. Sax, *Reliable Data Protocol RFC 908*, July 1984.

[8] E. Kohler, M. Handley, S. Floyd, *Datagram Congestion Control Protocol (DCCP) RFC 4340*, March 2006.

[9] R. C. Durst, G. J. Miller, E. J. Travis, *TCP extensions for space communications*, in MobiCom 1996, Rye, New York, USA.

[10] Object Management Group (OMG), *Data Distribution Service for Real-time Systems, v1.2*, January 2007.

[11] Sun Microsystems, *Java Message Service Specification - version 1.1*, March 2002.